

CENG 232

Logic Design

Spring '2019-2020

Lab 4

Part 1 Due Date: 5 May 2020, 23:59
Part 2 Due Date: 12 May 2020, 23:59
No late submissions

1 Part 1: Single-Error Correcting Memory: SEC_MEM (40 pts)

In this part of the assignment, you are going to implement a Single-Error Correcting Memory module called SEC_MEM which uses hamming code to detect and correct 1 bit errors in case of an internal data corruption while writing to the memory.

SEC_MEM module contains 3 modules, which are CheckParity, AddParity and RAM. The overall structure is given in Figure 1.

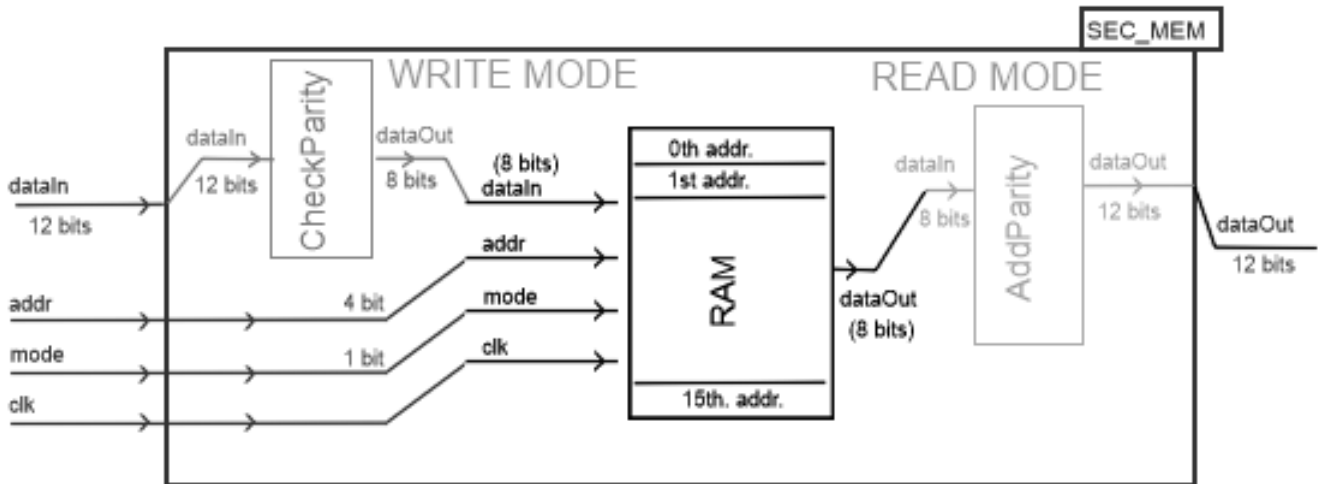


Figure 1: Illustration of the modules.

1.1 AddParity Module

Use the following Verilog definition for the module:

```
module AddParity(  
input [1:8] dataIn,  
output reg [1:12] dataOut  
);
```

In this module, 4 parity bits (P1, P2, P4 and P8) are added to the 8-bit data word (dataIn), forming a new word of 8+4 bits and store the result in dataOut. You should take into consideration the below specifications:

- Bit positions in dataIn are numbered from 1 to 8.

bit position	1	2	3	4	5	6	7	8
dataIn	d1	d2	d3	d4	d5	d6	d7	d8

- Bit positions in dataOut are numbered from 1 to 12. In the table below, p1,p2,p4,p8 are the parity bits and d1-d8 are the bits of dataIn.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
dataOut	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8

- Each parity bit is calculated as follows:
 - p1 = XOR of bits (3, 5, 7, 9, 11) : E.g. $d1 \oplus d2 \oplus d4 \oplus d5 \oplus d7$
 - p2 = XOR of bits (3, 6, 7, 10, 11) : E.g. $d1 \oplus d3 \oplus d4 \oplus d6 \oplus d7$
 - p4 = XOR of bits (5, 6, 7, 12) : E.g. $d2 \oplus d3 \oplus d4 \oplus d8$
 - p8 = XOR of bits (9, 10, 11, 12) : E.g. $d5 \oplus d6 \oplus d7 \oplus d8$

- A sample input and output is given below:

bit position	1	2	3	4	5	6	7	8	Bit position	1	2	3	4	5	6	7	8	9	10	11	12
dataIn	1	1	0	0	0	1	0	0	dataOut	0	0	1	1	1	0	0	1	0	1	0	0

1.2 CheckParity Module

Use the following Verilog definition for the module:

```
module CheckParity(
input [1:12] dataIn,
output reg [1:8] dataOut
);
```

In this module, 12 bit dataIn which contains data and parity bits is checked for a 1 bit error. After correcting the error (if necessary) 8 bit corrected data word is stored in dataOut. You should take into consideration the below specifications:

- Bit positions in dataIn are numbered from 1 to 12.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
dataIn	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8

- Bit positions in dataOut are numbered from 1 to 8.

bit position	1	2	3	4	5	6	7	8
dataOut	d1	d2	d3	d4	d5	d6	d7	d8

- The location of the error in dataIn is pointed by $C = C_8C_4C_2C_1$ and evaluated as follows:
 - C_1 = XOR of dataIn bits (1, 3, 5, 7, 9, 11)
 - C_2 = XOR of dataIn bits (2, 3, 6, 7, 10, 11)
 - C_4 = XOR of dataIn bits (4, 5, 6, 7, 12)
 - C_8 = XOR of dataIn bits (8, 9, 10, 11, 12)
- $C=0000$ indicates that no error has occurred. If $C \neq 0$, then the 4-bit binary number formed by $C_8C_4C_2C_1$ gives the position of the erroneous bit. **After pointing the error location, the value of the erroneous bit should be inverted to correct the error.**
- A sample input and output is given below (The erroneous bits are coloured red):

dataIn	Description	dataOut
0 0 1 1 1 0 0 1 0 1 0 0	$C = 0000$ (no error)	1 1 0 0 0 1 0 0
1 0 1 1 1 0 0 1 0 1 0 0	$C = 0001$ (error in bit 1)	1 1 0 0 0 1 0 0
0 0 1 1 0 0 0 1 0 1 0 0	$C = 0101$ (error in bit 5)	1 1 0 0 0 1 0 0

1.3 RAM module

Use the following lines as the port definition of the RAM Module:

```
module RAM(  
input [7:0] dataIn, //0:read, 1:write  
input clk,  
input mode,  
input [3:0] addr,  
output reg [7:0] dataOut);
```

The RAM will use 4-bit addresses. Each address will contain 8 bits of data. You should take into consideration the below specifications:

- Initially, the values of all RAM registers will be 0.
- mode(1 bit):
 - **0 - read mode:** When addr value is given as i , the value stored in the i th index of the RAM will be returned in dataOut when the positive edge clk pulse comes.
 - **1 - write mode:** 8-bit dataIn will be stored to the addr location of the RAM when the positive edge clk pulse comes.

1.4 SEC_MEM module

This is the upper module, in which inputs and outputs of other modules are defined. The inputs of this module; dataIn, clk, mode, address and dataOut are distributed to CheckParity, RAM and AddParity modules.

The module definition for the main module is below:

```
module SEC_MEM(  
input [1:12] dataIn,  
input clk,  
input mode, //0:read, 1:write  
input [3:0] addr,  
output [1:12] dataOut);
```

SEC_MEM has already been implemented by us; hence, **you should not implement this module.**

RAM, AddParity and CheckParity modules will be implemented by yourselves.

1.5 Hint

- For more information about hamming code, please check Chapter 7.4 “ERROR DETECTION AND CORRECTION” on your textbook.

1.6 Deliverables

- Implement both modules in a single Verilog file: **lab4.1.v**. Do NOT submit your testbenches. You can share your testbenches on the ODTUClass discussion.
- Submit the file through the ODTUClass system before the given deadline. **3 May 2020, 23:59**
- This is an individual work, any kind of cheating is not allowed.

2 Part 2: Pharmacy Entry System (60 pts)

In this part of the assignment, you are going to implement an entry/exit system for the pharmacy in our campus. The system is summarized as follows:



1. Because of coronavirus, the maximum capacity of the pharmacy is restricted to 10.
2. We store studentIDs and check-in times of the students entering the pharmacy.
3. The memory to store the studentID and check-in time values is implemented as a min-heap structure.
4. The min-heap is arranged according to the check-in time of the students. The student with minimum check-in time should be placed at the root.
5. The students are not inserted to the system by check-in time order. E.g. a record with check-in time 10 can be inserted before a record with check-in time 3. When a new student data is inserted to the memory, you should maintain the min-heap structure.
6. The system allows to list the studentIDs in the memory. The studentIDs are listed one by one in every positive edge of the clock cycle. Display order of the studentIDs should be the order of the student records in min-heap structure.
7. Initially, you should set each record of the memory to “00000”s.
8. Assume that there is no student with studentID=“00000”.

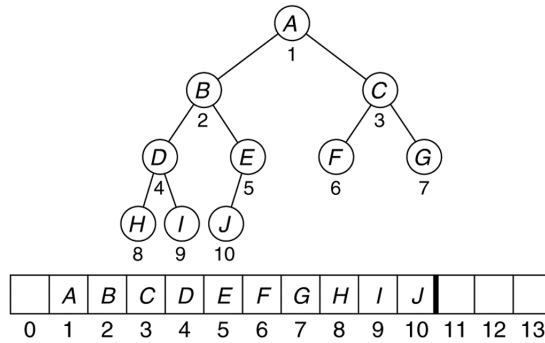
2.1 Module Definiton

There is only one module in the system called PharmacyMem module. The module definition is given below;

```
module PharmacyMem
(
input [4:0] studentID,
input clk,
input [1:0] mode, // 0:list ,2:checkIn 3:delete
input [7:0] checkInTime,
output reg [4:0] listOutput,
output reg listBusy,
output reg ready
);
```

You should take into consideration the below specifications:

1. studentID represents 5-bit code in the smart card of a student.
2. clk is the clock input for the module. The system is triggered by the **positive edge** of the clock.
3. mode is used to make different operations on the system:
 - mode = ‘10’: CheckIn Mode
 - Students can enter the pharmacy when the system is in CheckIn Mode.
 - If pharmacy capacity is full then the student is “not” allowed to enter the pharmacy.
 - If a student is allowed to enter the pharmacy (there is enough capacity), the system will record the studentID in his/ her smart card and check-in time into the heap memory.
 - mode = ‘11’: Delete minimum
 - In this mode, delete the student with minimum check-in time from the memory.
 - mode = ‘00’: List mode
 - In this mode, list the studentIDs in listOutput one by one in every positive edge of the clock. The list order should be the order of the records in the min-heap.



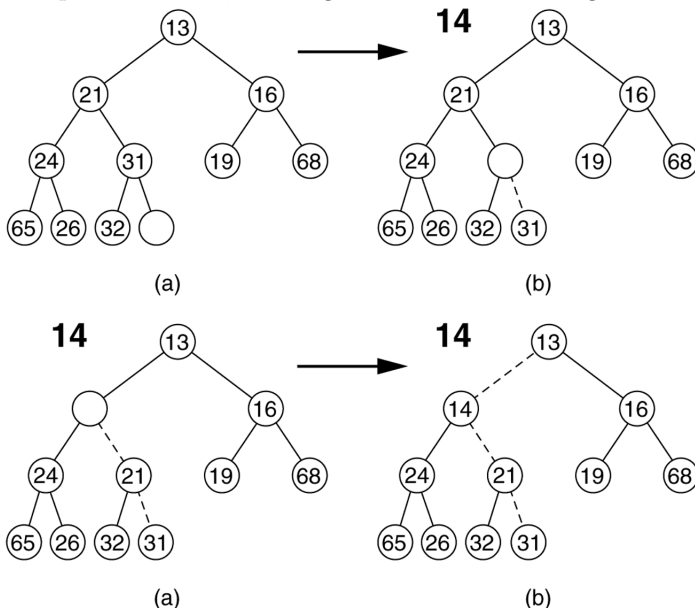
- The output should not list empty records.
 - While displaying the records, it should be indicated if the system is in the List Mode(listBusy=1).
 - 1 clock after showing all records, it should be indicated if the system has finished showing all records. (ready=1, listBusy=0)
 - If system is ready (finished showing all records) then after 1 clock cycle, it will start from the beginning. (listBusy=1 and ready=0)
4. listOutput displays the studentID records in the memory sequentially each time clk pulse is given.
 5. listBusy indicates whether the system is currently displaying a studentID.
 - listBusy =1: System is in list mode and a studentID is listed in listOutput.
 - listBusy = 0: otherwise
 6. ready indicates whether list mode finishes showing all records. We will check this only if the system is in list mode.
 - ready =1: System finishes showing all records
 - ready = 0: otherwise

2.2 Heap operations

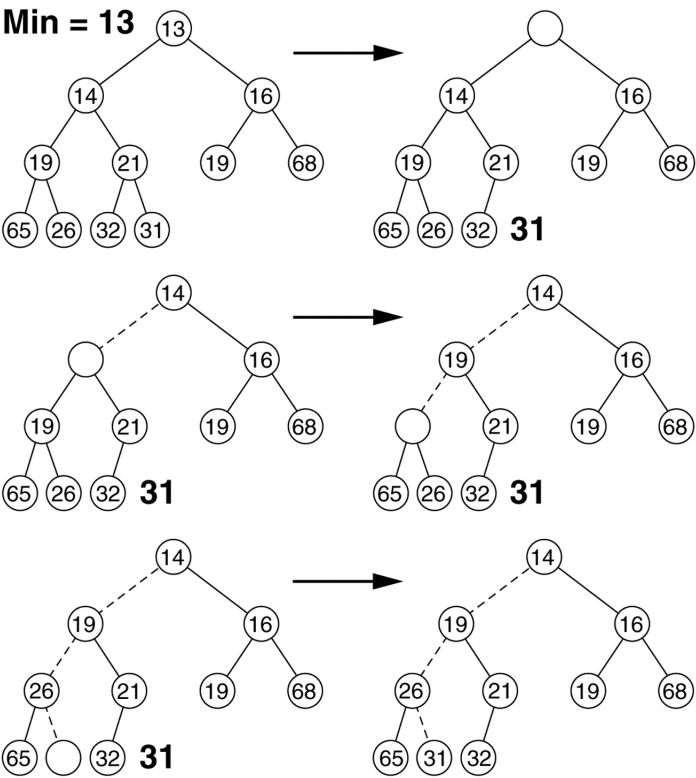
Ceng213 Data Structures lecture slides were used as a reference for the following insert/delete heap operations. In this homework you are supposed to implement two heap operations; **insert** and **delete minimum**. Each of these operations are supposed to be done in one clock cycle.

2.2.1 Insert Operation in Heap

Attempt to insert 14, creating the hole and bubbling the hole up:



2.2.2 Delete Minimum in Heap



2.3 Sample Current State / Next State

Initial Values

Pharmacy Memory	
studentID	checkInTime
00000	0
00000	0
00000	0
00000	0
00000	0
00000	0
00000	0
00000	0
00000	0
00000	0

listOutput	listBusy
No output	0

Between current state and next state, there will be one clk cycle.

1. A student having studentID “01001” checks in the pharmacy at time 5.

Current State

Pharmacy Memory		Heap Structure
studentID	CheckInTime	
00000	0	EMPTY
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	

studentID	mode	listOutput	listBusy
01001	10	No output	0


Next State

Pharmacy Memory		Heap Structure
studentID	CheckInTime	
01001	5	5
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	

listOutput	listBusy
No output	0

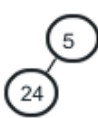

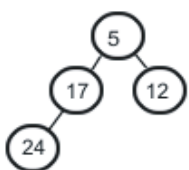
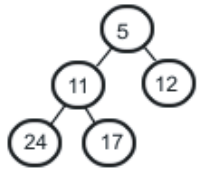
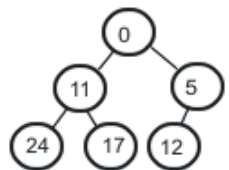
2. 5 students, 01010, 01101, 00100, 01000 and 10010 want to enter the pharmacy respectively.
 - studentID:01010, checkInTime:24
 - studentID:01101, checkInTime:12
 - studentID:00100, checkInTime:17
 - studentID:01000, checkInTime:11
 - studentID:10010, checkInTime:0

Current State

Pharmacy Memory		Heap Structure
studentID	CheckInTime	
01001	5	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	
00000	0	

studentID	mode	listOutput	listBusy
01010	10	No output	0

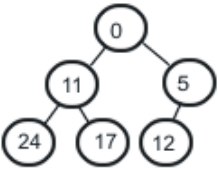
Next State

Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.	
studentID	Time	studentID	Time	studentID	Time	studentID	Time	studentID	Time
01001	5	01001	5	01001	5	01001	5	10010	0
01010	24	01010	24	00100	17	01000	11	01000	11
		01101	12	01101	12	01101	12	01001	5
				01010	24	01010	24	01010	24
						00100	17	00100	17
								01101	12
Ins. 24		Ins. 12		Ins. 17		Ins. 11		Ins. 0	
Heap Structure		Heap Structure		Heap Structure		Heap Structure		Heap Structure	
									

listOutput	listBusy
No output	0

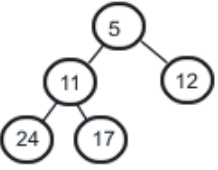




3. Delete min. operation (5 times)

Current State

Pharmacy Memory	
StudentID	CheckInTime
10010	0
01000	11
01001	5
01010	24
00100	17
01101	12
Heap Structure	
	

mode	listOutput	listBusy
11	No output	0

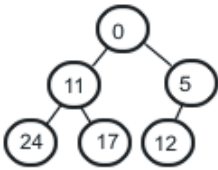
Next State

Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.		Pharmacy Mem.	
studentID	time	studentID	time	studentID	time	studentID	time	studentID	time
01001	5	01000	11	01101	12	00100	17	01010	24
01000	11	00100	17	00100	17	01010	24		
01101	12	01101	12	01010	24				
01010	24	01010	24						
00100	17								
1st. del. Min.		2nd. Del. Min.		3rd. del. Min.		4th. del. Min.		5th. del. min	
Heap Structure		Heap Structure		Heap Structure		Heap Structure		Heap Structure	
									

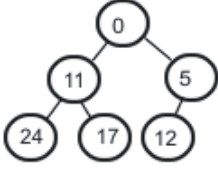
listOutput	listBusy
No output	0

4. List Mode (mode=00)

Current State

Pharmacy Memory		Heap Structure	
studentID	CheckInTime		
10010	0		
01000	11		
01001	5		
01010	24		
00100	17		
01101	12		
mode	listOutput	listBusy	Ready
00	No output	0	1

Next State

Pharmacy Memory		Heap Structure	
studentID	CheckInTime		
10010	0		
01000	11		
01001	5		
01010	24		
00100	17		
01101	12		
listOutput	listBusy	Ready	
10010	1	0	

- After every positive edge clk pulse, the listOutput will display the next record. (01000, 01001, 01010, 00100, 01101).
- Please note that, the order of the studentIDs is the order of the records in the min-heap memory. (They should not be ordered according to checkInTime.)
- 1 clock after all the records are displayed, it will set ready to 1 and listBusy to 0.
- 1 Clock cycle after ready is 1, it will start displaying from the beginning (setting ready to 0 and listBusy to 1). (The output should not list empty records)

2.4 Hint

You can check insert & delete operations using link given below:

<https://www.cs.usfca.edu/~galles/visualization/Heap.html>

2.5 Deliverables

- Implement both modules in a single Verilog file: **lab4.2.v**. Do NOT submit your testbenches. You can share your testbenches on the ODTUClass discussion.
- Submit the file through the ODTUClass system before the given deadline. **12 May 2020, 23:59**
- This is an individual work, any kind of cheating is not allowed.